VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

# Nasazení stavového filtru síťového provozu pro systém Virtlab

# Statefull Firewall for Virtlab System

2011 Václav Dener

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 17.dubna 2011 .....

Na tomto místě bych rád poděkoval panu Ing. Martinu Milatovi, za pomoc na konzultacích k této práci a panu Ing. Petru Jaššovi za podporu a pomoc při seznamování s iptables.

## Abstrakt

Cílem práce je vytvořit stavový paketový filtr, pro potřeby sítě Virtlab a také prostudování nástroje iptables a jeho možných grafických a programových rozšíření. V práci jsou shrnuty poznatky získanné při této činnost a výsledeky z ní plynoucí.

**Klíčová slova:** Virtlab, paketový filtr, stavový paketový filtr, netfilter, iptables, linux

## Abstract

The goal of this thesis is to create a state packet filter for the needs of the Virtlab and also conduct research of the iptables tool and its user intrface and programmatic extensions. In the paper, there are summarized the information gained during the research and its results.

**Keywords:** Virtlab, packet filter, Statefull Firewall, netfilter, iptables, linux

# Obsah

# Seznam obrázků

# Seznam výpisů zdrojového kódu

# 1 Úvod

Cílem této práce bylo vytvoření firewallu, pomocí linuxového programu IPTABLES. Teno firewall by měl být nasazen v prostředí virtuální síťové laboratoře, provozované VŠB-TUO nazývanou VirtLab. V tomto případě by se mělo jednat o stavový firewall na vnějším rozhraní sítě VirtLab.

# 2 Vysvětlení pojmu firewall

Firewall je síťový prvek, který slouží k zabezpečování síťového provozu mezi sítěmi a klienty s různou úrovní důvěryhodnosti a zabezpečením. Firewall lze také popsat jako kontrolní bod v síti, kterým procházejí pakety a jsou zde definována pravidla pro komunikaci mezi sítěmi, které jsou jím od sebe odděleny. Dříve k tvorbě těchto pravidel plně dostačovala znalost cílové a zdrojové ip adresy a portu, na kterém se komunikovalo.

V dnešní době se využívá kromě těchto informací, také informací o stavu spojení, znalost kontrolovaných paketů a případně prvky Intrusion Detection System (Systém detekce průniku)

Typy firewallu:

- Paketový filtry
- Aplikační brány
- Stavové paketové filtry
- Stavové paketové filtry s kontrolou IDS

# 3 Typy firewallu

## 3.1 Paketový filtry

Jednou z nejjednodušších a nejstarších používaných možností nastavení firewallu, je paketový filtr, kde jsou jednotlivé pakety třízeny podle ip adresy a portu zdroje a cíle, tato kontrola se provádí na třetí a čtvrté vrstvě ISO-OSI modelu. U tohoto typu firewallu se však vyskytuje nebezpečí, u složitějších protokolů jako například FTP. Jelikož v tomto nastavenínení vůbec kontrolován obsah přenášených paketů, mohou nekteré protokoly otevřít námi nepředpokládané porty a směry, které mohou být využity pro nás nežádanými protokoly.

Navzdory těmto nevýhodám je tento firewall díky své jednoduchosti, dosahovat u tohoto řešení vysoké rychlosti kontroly paketů. Proto je toto řešení používáno v místech, kde není vyžadována dukladná kontrola provozu a je kladen důraz na vysokou rychlost průchodu velkého objemu dat přez firewall.

Mezi představitele paketových filtrů patří access listy dále jen ACL, které jsou využívány v systémech IOS nebo JunOS, nebo také například ipchains.

## 3.2 Aplikační brána

Tento typ firewallu oproti předchozímu modelu plně odděluje sítě, mezi kterými je vybudován. Celý tento model funguje na principu klient-server, kde se ze zdrojové adresy iniciátora spojení odešlou data na aplikační bránu, někdy také nazývanou proxy firewall, kterým je spracována a na základě požadavku klienta, je vytvořeno nové spojení na cílový server. Data která pak firewall obdrží od cílového serveru, opět stejným způsobem předá zpět klientovi. Celý tento proces se provádí na sedmé aplikační vrstvě OSI modelu, odtud také název aplikační brána. Využívání tohoto modelu má pak za následek to, že klient nevidí přímo adresu cílového serveru, ale vnější adresu aplikační brány, z čehož plyne že na aplikační bráně dochazí k překladu adres (NAT).

Nevýhodou tohoto typu firewallu je jeho vysoká náročnost na hardware, z které se odvíjí řádově vyšší časová náročnost na zpracování paketu, a taky vyšší latence než u paketového filtru. Pro každý použitý protokol je nutné napsání speciální proxy, proto je také omezení v počtu kontrolovaných protokolů. Mezi výhody naopak patří vyšší zabezpečení složitějších protokolů, a také kompletní oddělení jednotlivých sítí.

## 3.3 Stavový paketový filtr

Stavový paketový filtr je vylepšenou verzí běžného paketového filtru. Dochází v něm k jednoduché kontrole příchozích paketů, na základě pravidel vztahujících se na ip adresy

a po rty. Na rozdíl od klasického paketového filtru, se zde ukládají také informace o povolených spojeních a ty jsou pak využívány při rozhodovacím procesu firewallu. Díky tomuto je stavový paketový filter schopen zjišťovat, zda právě procházející paket patří do již otevřeného spojení, nebo zda musí znovu projít rozhodovacím procesem firewallu, tímto se velmi zrychluje zpracování paketu, a také velmi zjednodušuje konfigu-

raci paketového filtru, protože již do pravidel není nutno nastavovat akci pro odpovědní směr komunikace, ten je vytvořen automaticky. Například u protokolu FTP je třeba nastavit pouze povolení odchozího směru a komunikace na portu 21, která je standartně využívána pro komunikace u protokolu FTP. Odpovední směr na portu 21 je povolen automaticky, zároveň je rovněž povolena komunikace na portu 22, který je používán pro odesílání souborů pomocí protokolu FTP.

Mezi výhody tohoto typu firewallu patří kromě vysoké rychlosti zpracování paketu, což je dáno jednak relativní jednoduchostí paketového filtru a taky využitím uložených informací o otevřených spojeních. Další výhodou je snadnější konfigurace oproti předešlým firewallům, z důvodu odpadající nutnosti konfigurace pravidel pro odpovědní směry, tím pádem i nižší náchylnost k chybě při pravidelné údržbě.

Příkladem pužití stavových paketových filtrů je například Cisco IOS firewall nebo v prostředí linuxu program iptables.

## 3.4 Stavový paketový filtry s kontrolou protokolů a IDS

[**?**] U těchto typů firewallu dochází kromě kontroly paketu na základě uživatelem vytvořených pravidel a díky conection trackingu také kontrole složitějších dobře známých protoloů, jako například FTP, tak predevším přináší oproti předchozým variantám IDS (Deep Inspection nebo Application Intelligence podle výrobce firewallu), které umožouje kontrolovat procházející spojení až na úroveň korektnosti procházejících dat známých protokolů i aplikací. Mohou tak například zakázat průchod http spojení, v němž objeví indikátory, že se nejedná o požadavek na WWW server, ale tunelování jiného protokolu, což často využívají klienti P2P sítí (ICQ, gnutella, napster, apod.), nebo když data v hlavičce e-mailu nesplňují požadavky RFC apod.

Nejnověji se do firewallů integrují tzv. in-line IDS (Intrusion Detection Systems – systémy pro detekci útoků). Tyto systémy pracují podobně jako antiviry a pomocí databáze signatur a heuristické analýzy jsou schopny odhalit vzorce útoků i ve zdánlivě nesouvisejících pokusech o spojení, např. skenování adresního rozsahu, rozsahu portů, známé signatury útoků uvnitř povolených spojení apod.

Výhodou těchto systémů je vysoká úroveň bezpečnosti kontroly procházejících protokolů při zachování relativně snadné konfigurace, poměrně vysoká rychlost kontroly ve srovnání s aplikačními branami, nicméně je znát významné zpomalení (zhruba o třetinu až polovinu) proti stavovým paketovým filtrům.

Nevýhodou je zejména to, že z hlediska bezpečnosti designu je základním pravidlem bezpečnosti udržovat bezpečnostní systémy co nejjednodušší a nejmenší. Tyto typy firewallů integrují obrovské množství funkcionality a zvyšují tak pravděpodobnost, že v některé části jejich kódu bude zneužitelná chyba, která povede ke kompromitování celého systému.

# 4 Netfilter/iptables

Iptables je linuxový program, který slouží v příkazovém řádku pro ovládání části linuxového jádra Netfilter, který se stará o všechna příchozí, odchozí i procházející spojení, jinými slovy se pomocí iptables se tvoří pravidla pro stavový firewall, obsažený ve většině systémů založených na linuxu. Nástroj iptables je používán od verze jádra 2.4, kde nahradil v predchozích verzích obsažený ipchains a ipfwadm. Iptables manipuluje s Xtables, které následně obsahují řetězce v nichž jsou jednotlivá pravidla pro paketový filtr.

## 4.1 Tabulky iptables

Iptables se skládají ze 4 základních tabulek, které následně obsahují jednotlivé řetězce.

Typy tabulek firewallu

- FILTER Implicitně je nastavena tato tabulka. Tabulka obsahuje řetězce INPUT, OUTPUT a FORWARD
- NAT Tabulka obsahuje řetězce POSTROUTING a PREROUTING. V této tabulce dochází k modifikaci paketu neboli NAT, které již prošli routovací tabulkou.
- MANGLE Obsahuje tyto tabulky INPUT, OUTPUT, FORWARD, PREROUTING a POSTROUTING. Nacházejí se v ní sady pravidel pro úpravu hlavičky paketu, například manipulace s TTL, TOS flagy.
- RAW řetězce PREROUTING, OUTPUT. Tato tabulka převážně obsahuje pravidla aplikovaná na spojení, kteřé nechceme zanášet do seznamu conntrack, tato tabulka je volána ještě před zápisem spojení do tohoto seznamu.

## 4.2 Řetězce iptables

řetězce v iptables jsou uspořádané soubory jednotlivých pravidel. U základních předem definovaných řetězců, například v tabulce FILTER jsou to OUTPUT, INPUT a FORWARD se navíc definují ještě politiky, což jsou pravidla platící obecně pro celý řetězec a používají se jako výchozí pravidlo pokud paket nevyhovuje žádnému pravidlu v řetězci.

řetězec INPUT - tímto řetězcem procházejí všechny pakety, které mají v hlavičce paketu jako cílovou adresu zapsánu ip adresu serveru na kterém je firewall spuštěn a není tímto strojem routován dále.

řetězec OUTPUT - do tohoto řetězce spadají pravidla ovlivňující pakety v jejichž hlavičce je zdrojová adresa shodná s adresou námi provozovaného stroje, takže tímto řetězcem procházejí všechny odchozí spojení.

řetězec FORWARD - v tomot řetězci jsou obsažena pravidla zabívající se pakety, které přez prvek na němž je paketový filtr provozován pouze procházejí, neboli je tímto síťovým prvkem pouze routován dále do sítě, takže je o něm záznam v routovací tabulce.

## 4.3 Tvorba vlastních řetezců

Iptables poskytuje i možnost vytváření vlastních řetězců. Tato možnost napomáhá k lepší škálovatelnosti a přehlednosti navrhovaného firewallu. Mimo tyto výhody také velmi ulehčuje hardwaru firewallu, to z toho důvodu že pravidla v jednotlivích řetězcích jsou kontrolována sekvenčně, proto čím méně pravidel je v řetězci, tim je zpracování tohoto řetězce rychlejší. U vetšího možství pravidel v jednom řetězci, případném špatném uspořádání pravidel v řetězci ( velmi frekventovaná pravidla jsou uvedena mezi posledními a proto je nutné projít i méně používaná pravidla, která jsou uvedena před nimi) zatěžuje neúměrným způsobem hardware a může mít za následek snížení propustnosti datového toku. S využitím vlastních řetězců lze tuto vlastnost omezit, tím že série pravidel pro jednotlivé situace umísťíme do samostatných řetězců, na které se pak odkazujeme pravidly z jednoho z defaultních řetězců jako INPUT, OUTPU, FORWARD atd., čímž se rapidně sníží počet kontrolovaných pravidel.

## 4.4 Typy akcí firewallu

Snytaxe pravidel v ip tables:

```
iptables [tabulka] [akce] [retezec] [ ip_cst ] [match] [cl ] [ cl_info ]
```

Výpis 1: Snytaxe pravidel v ip tables

Pokud tedy paket vyhovuje pravidlu ve správné tabulce a řetězci, je možno zjišťovat zda paket vyhovuje dalším informacím obsaženým v hlavičce každého paketu, jako jsou například zdrojová,cíová adresa nebo zdrojový,cílový port případně síťové rozhraní zařízení.

Pokud těmto parametrům paket nevyhovuje postupuje tento paket k dalšímu pravidlu v řetězci, připadně pokud již žádné další pravidlo není je na něj aplikována politika daného řetězce, v opačném případě je toto pravidla na paket aplikováno a vykoná akci zadanou tvůrcem firewallu. Těchto akcí je několik:

- ACCEPT Paket je propuštěn skrze firewall.
- DROP Prosté zahození paketu.
- REJECT Zahození paketu, s tím rozdílem, že zpět na zdrojovou adresu, ze které paket prišel, je odeslána ICMP zpráva o zahození tohoto paketu.
- LOG Při této akci dochází k tomu, že paket po vykonání tohoto pravidla neopustí firewall jako u předchozích akcí, ale pokračuje dále v procházení řetězce. Toto pravidlo bylo dodáno do pozdější verze kernelu. Při této akci je do běžného logu systému zapsána hlavička kontrolovaného paketu.

## 4.5 Match extension

Pro lepší specifikaci jednotlivých pravidel se využívá rozšířených možností iptables, které je možno zapnou v jádru systému. Těchto modulů je poměrně velké množství, mezi často

používané patří například rozšíření limity, které lze použít k omezení počtu vytvořených spojení, nebo například početu zápisů do logu za určitý časový úsek. V tomto případě pokud je shodných zápisů více, lze i určit počet zápisů z vytvořeného spojení.

Mezi další rozšíření patří platnost pravidla pouze po určitý časový úsek. Dalším podstatným modulem, který převážně odlišuje iptables od jeho předchůdce ipchain, je modul tvořící z iptables stavový firewall, kde můžeme určit stav paketů, jako NEW, REALETED, FIN, INVALID, které bude pravidlo využívat při rozhodování aplikačního filtru.

Jako další příklad rozšíření lze uvést modul accont, který nám umožouje zaznamenávání trafficu na síťovém rozhraní firewallu, díky následným úpravám lze získaná data použít k vytvoření grufu zátěže firewallu.

## 4.6 Network Address Translation

Network Address Translation neboli NAT se používá při potřebě routru upravovat hlavičky paketů, které jím prochází. Tento proces se v iptables odehrává v tabulce NAT v řetězcích PREROUTING kde se provádí DNAT neboli změny cílové adresy či portu případně přesměrování na jinný port.

Obdobně je tomu u SNAT, který je provozován v tabulce NAT v řetězci POSTROUTING a podobně jako v předchozím případě zde dochází k modifikaci zdrojové ipadres a portů paketu.

Technika NAT se v dnešní době začínajícího nedostatku volných, veřejných ip adres používá k takzvané maškarádě, kde je za routrem více adres a pomocí NAT jsou pak hlavičky paketů upravovány tak, aby se všechny adresy navenek z internetu jevily jako jedna veřejná adresa. Toto nastavení přináší významnou úsporu ip adres, avšak nese sebou jak zvýšené nároky na hardware routru, tak také problémy určitých aplikací při komunikaci s vnější sítí.

# 5 Ostatní nástroje pro správu iptables

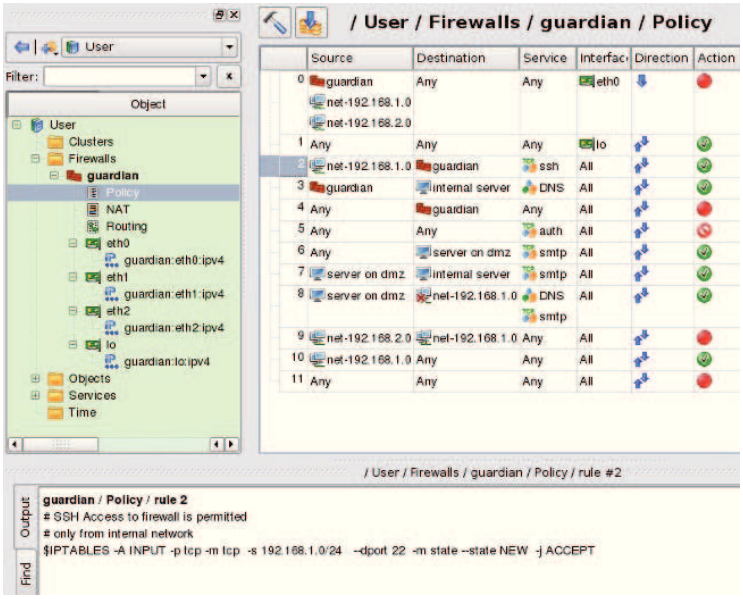
## 5.1 FwBuilder

FwBuilder je graficko uživatelské rozhraní generující skripty pro tvorbu firewallu nad iptables. Kromě pravidel pro firewall má tento program ve své funkcionalitě také obsaženu správu routovacích a NATovacích pravidel. Tento software je šířen pod licencí GNU/-GPL, takže je k dostání zdarma.

Je obsažen v balíčkovacích systémech většiny běžných distribucí systému Linux. Kromě Linuxu lze toto GUI provozovat jak na systému, Windows tak také na OS X, OpenBSD, FreeBSD díky této univerzálnosti je hojně využíván a usnadňuje přenášení nastavení mezi těmito systémy, pomocí funkce import/export.

Výhodou Firewall Builderu je relativní přehlednost jednotlivých pravidel firewallu. Kromě této výhody poskytuje tento program také automatické doplňování reverzních pravidel v paketovém filtru, tato schopnost se hodí hlavně v případech, kdy je z nějakého důvodu vypnut connection tracking.

Také odpadá nutná znalost jednotlivých, dobře známých portů pro použité protokoly, tyto porty jsou již v programu obsaženy a slovně popsány, z čehož plyne zjednoduššení konfigurace a pravidelné údržby firewallu a tudíž nižší náchylnost ke vzniku chyby v důsledku lidského faktoru.

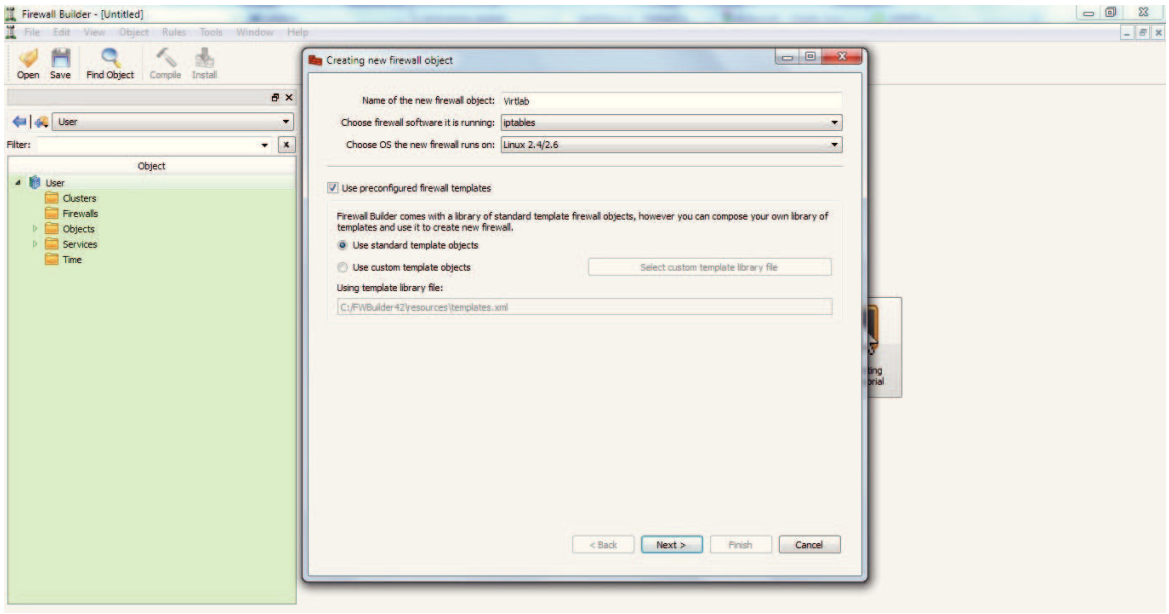



Obrázek 1: GUI-Firewall builder

### 5.1.1 Postup tvorby firewallu ve fwbuildru

Jednou z prvních věcí při tvorbě nového firewallu v prostředí fwbuildru, je nutnost výběru softwaru na kterém firewall poběží, na výběr je velké množství softwaru od Cisco IOS ACL, přez starší ipfilter až po novější iptables. Dalším krokem je výběr operačního systému na kterém firewall běží. Na výběr jsou nejrůznější OS jako třeba klasické Windows pak také Linux, ale i méně známé jako HP ProCurve nebo Sveasoft.

Následujícím krokem a zároveň velký ulehčením pro začínající administratory je možnost výběru templatu, kde je na výběr z několika nejčastěji se vyskytujících konfigurací, pro které jsou připravena jednotlivá základní pravidla jako například antispoof pravidla atd.

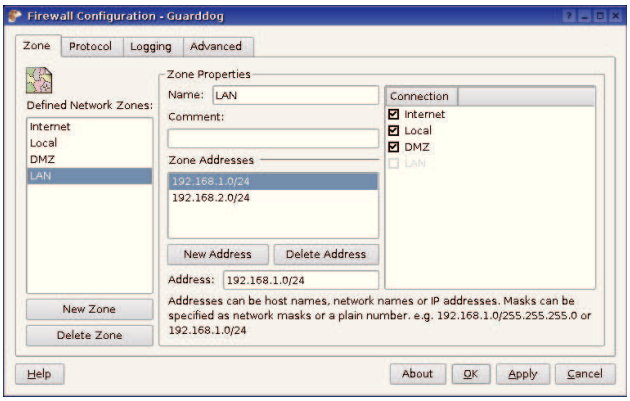



Obrázek 2: Fwbuilder-základní nastavení

Po nastavení základních hodnot se již můžeme přejít k tvorbě jednolivých pravidel firewallu. K pomoci při tvorbě pravidel slouží systém knihoven ve kterých jsou uspořádány jednotlivé služby fungující na dobřeznámých portech, tak také rozsahy ip adress zaregistrované společenstvím IANA pro výhradní použití jako domácí sítě a jiné. Sami uživatelé si mohou tvořit vlastní knihovny, které se dají přenášet mezi jednotlivimy systémy. Po vytvoření sady pravidel firewallu se může přejít ke kompilaci pravidel a vytvoření skriptu, který se podle zvoleného softwaru a operačního systému může použít na daném stroji. U kompilace lze v nastavení také zapnout debuging ke snadnějšímu odhalování chyb. FWbuilder obsahuje i nástroj k automatickému zavedení zkompilovaného souboru na zvolený server, který je připojený k síti.

## 5.2 Guarddog

Software 2.4 a 2.6 jádru a je šířeno pod licencí GNU/GPL a tudíž zdarma. Výhodou tohoto GUI je velké množství přednastavených vzorů, které pak může i méně zkušený uživatel upravit a vytvořit dobře fungující firewall. Další výhodou tohoto softwaru je možnost importu/exportu vytvořeného nastavení a mezi další výhody patří kvalitní technická dokumentace od tvůrců aplikace.




Obrázek 3: GUI-Guarddog

# 6 Virtual Network Lab

## 6.1 Obecné informace o Virtual Network Lab

[?] Smyslem projektu Virtlab je zpřístupnit laboratorní prvky pro praktickou výuku počítačových sítí vzdáleně prostřednictvím Internetu. Studenti si mohou pomocí WWW rozhraní rezervovat laboratorní prvky na určitý časový interval a následně k nim přistupovat pomocí běžného WWW prohlížeče s podporou Java appletů. Propojení laboratorních prvků se uskuteční automaticky podle výběru konkrétní úlohy ze souboru nabízených laboratorních úloh, nebo si student může zadat svou vlastní topologii. Systém nyní dovoluje spolupráci více lokalit vzájemně sdílejících síťové prvky a realizaci virtuálních síťových topologií přes Internet. Fyzické síťové prvky nutné pro vytvoření studentem vybrané topologie jsou v době rezervace vyhledávány dynamicky ve všech lokalitách.

### 6.1.1 Historie vzniku Virtlab

[?] Myšlenka virtuální laboratoře se vyvinula z potřeby poskytnout možnost řešení praktických laboratorních úloh studentům kombinovaného studia a také zpřístupnit jinak méně využité a často nákladné laboratorní síťové prvky pro samostatnou práci v časech mimo výuku.

Její vznik inicioval Petr Grygárek a postupně ji realizuje s pomocí diplomantů, zejména inženýrského studia, na katedře informatiky. Základní koncepce systému byla definována v roce 2005 v diplomové práci Pavla Němce, který implementoval i základní prototyp aplikace. O rok později prototyp rozšířil formou diplomové práce Roman Kubín, který implementoval bezpečnostní prvky, podporu práce studentů s tutorem a v návaznosti na Automatizovaný systém správy síťových konfigurací (ASSSK a.k.a. Tatabazmek) vyvinutý v rámci diplomové práce Davida Seidla možnost definice vlastní topologie propojení síťových prvků podle přání studenta.

Koncepci automatizovaného systému pro spojování topologií poté Petr Grygárek zobecnil, aby bylo možné propojovat nejen sériové porty, ale i Ethernet porty včetně trunk spojů. Příslušné konfigurační skripty implementoval Jiří Dvořák, čímž vzniklo tzv. virtuální spojovací pole. Později byla s pomocí Tomáše Kučery do systému implementovány pracovní stanice simulované s použitím instancí User Mode Linux a za podpory Jiřího Dvořáka také virtuální směrovače Cisco 7200 realizované s použitím projektu DynaMIPS/DynaGen. Martin Milata následně nahradil simulaci stanic pomocí UML použitím XEN, který se ukázal se systémem Virtlab lépe integrovatelný (přístup na konzole pomocí čistého TCP spojení).

Diplomant Ing. Davida Seidla Petr Sedlář v roce 2007 reimplementoval ASSSK-1 s použitím FPGA (nové zařízení je nazýváno ASSSK-2), což zjednodušilo opakovanou realizaci a řešení výrazně zlevnilo. Ve stejné době dokončili diplomanti Jan Vavříček a Tomáš Hrabálek na distribuované verzi Virtlabu, která umožouje vytvářet rozsáhlejší topologie z laboratorních prvků umístěných v několika lokalitách připojených k Internetu a optimálně mapovat fyzické laboratorní prvky pro topologie úloh paralelně požadovaných různými studenty. Jiří Dvořák reimplementoval podle návrhu Petra Grygárka konfigurační

skripty spojovacího pole pro podporu více lokalit, což umožnilo vytváření virtuálních topologií přes Internet pomocí tunelovacího serveru Tomáše Hrabálka. Spojovací systém je nyní nazýván distribuovaným virtuálním spojovacím polem, které dovoluje tunelovat provoz mezi segmenty virtuálního spojovacího pole umístěnými v jednotlivých lokalitách.

## 6.2 Popis serverů sítě Virtlab

### 6.2.1 Virtlab - Rezervační server

[?] Rezervační server je démon, který běží vždy na jednom serveru v každé lokalitě distribuovaného systému a tedy v celém systému běží tolik instancí, kolik je definováno lokalit. Tento server má vždy určeno jméno své lokality, jména vzdálených lokalit a IP adresy jejich rezervačních serverů. Dále má ke každé lokalitě uveden seznam síťových prvků místní lokality, které může vzdáleným lokalitám poskytnout k zarezervování a to podle daného týdenního rozvrhu.

Když ovládací software chce zarezervovat určité prvky, pošle rezervačnímu serveru dotaz, které prvky jsou globálně v celém distribuovaném systému pro něj v určeném čase k dispozici. Rezervační server žádost zpracuje a odešle i do vzdálených lokalit. Každá lokalita má definován XML soubor s popisem vybavení místní laboratoře. Z něj rezervační server vybere nezarezervované a povolené prvky a odešle výsledný XML soubor tazateli. Rezervační server, který rozdistribuovával dotaz ovládacího software, poskládá všechny přijaté XML popisy dostupného vybavení do jediného souboru, který vrátí tazateli. Není-li k dispozici žádný síťový prvek, je vrácen platný soubor, ovšem bez zařízení.

Ovládací software XML soubor od Rezervačního serveru zpracuje, vybere z něj vhodné prvky, a může dále žádat o zarezervování seznamu síťových prvků v daném časovém rozmezí. K tomu vygeneruje globálně unikátní rezervační id ve tvaru 'celé číslo @ název lokality'. Rezervační server žádost rozdělí a pošle každé vzdálené lokalitě v žádosti jen ty prvky, které jí patří. Aby se eliminovaly konflikty, řekne místní rezervační server vzdáleným kolegům, aby onu rezervaci považovali za dočasnou. Povede-li se dočasná rezervace u všech žádaných lokalit, je všem rezervace potvrzena trvale. Může ovšem dojít k tomu, že si někdo před námi již daný síťový prvek zarezervoval a není možné rezervaci u některých z rezervačních serverů provést. Pak k potvrzení samozřejmě nedojde, po dané časové prodlevě dočasné rezervace vyprší a žádost o rezervaci selže.

### 6.2.2 Virtlab - Mazací server

[?] Uživatelé, během rezervované úlohy, mají možnost postupnými kroky modifikovat počáteční konfiguraci poskytnutých laboratorních zařízení. Vytvářená konfigurace je během řešení úlohy součástí celku, který však v okamžiku ukončení rezervace zaniká. Automaticky generovaná topologie, nad kterou uživatel svou úlohu řešil, je rozpojena a laboratorní prvky jsou poskytovány v budoucích rezervacích, kde je modifikovaná konfigurace nežádoucí.

Konfiguraci, kterou uživatel právě ukončené rezervace nemusel korektně odstranit, je zapotřebí, před znova zapojením daného síťového prvku do nové topologie, vymazat. Prvek je potřeba před jeho znovu použitím uvést do definovaného počátečního stavu, který bude dalšímu uživateli nejvíce vyhovovat.

Mazací server je zastoupen ve všech lokalitách a jeho primárním úkolem je odstraňování konfigurací, které vznikly na zařízeních dané lokality během řešení rezervovaných úloh. Mazací server vždy po skončení uživatelem zarezervovaného času uvede konfiguraci používaných zařízení do dříve definovaného, počátečního stavu.

### 6.2.3 Virtlab - Konzolový server

[**?**] Jeho úkolem je zprostředkovat přístup k sériovým nebo telnetovým konzolím síťových zařízení prostřednictvím jednoduchého protokolu nad TCP/IP. Je využíván Java Appletem, který běží ve webovém ovládacím rozhraní, což umožňuje uživateli jednoduchý přístup k síťovým zařízením. Zároveň slouží i jako proxy server, který zprostředkovává přístup k zařízením, která jsou připojena ke vzdáleným konzolovým serverům, kam nemá místní uživatel přímý přístup. Také prostřednictvím speciálního PHP skriptu ověřuje, jsou-li požadavky uživatele oprávněné a tím konzoly prvků zabezpečuje od neautorizovaného přístupu.

Zdrojový soubor server.c obsahuje hlavní funkci (main), která po úvodní inicializaci volá v nekonečném cyklu funkci obsluhuj$_klienta().Vefunkciobsluhujklienta()jsoupaknačteny4řádkyparam$

V případě úspěchu se nastaví alarm() a hned potom se dostane ke slovu funkce opendevfd(), která ověří, jestli zařízení již není používáno jiným apletem. Není-li, zamkne jej pro výlučný přístup a vrátí na něj deskriptor.

### 6.2.4 Virtlab - Tunelovací server

[**?**] Tunelovací server je démon běžící na zvláštním serveru, jež je součástí Distribuované virtuální laboratoře a jeho úkolem je zajistit propojení (bridge) různých VLAN mezi jednotlivými lokálními virtuálními laboratořemi a také různých VLAN v rámci jednoho ethernetového segmentu.

Dané ethernetové rozhraní serveru je připojeno trunk linkou k příslušnému přepínači (Cisco 3550) nebo i jinému zařízení, které podporuje standard IEEE 802.1q. Tunelovacímu serveru je nakonfigurována tabulka (tabulka přesměrování), ve které je uvedeno vždy VLAN ID rámce z lokání sítě, nové VLAN ID kam bude rámec poslán a ip adresa vzdáleného (nebo i lokálního) tunelovacího serveru. Tunelovací server přepne rozhraní s trunk linkou do promiskuitního módu, což umožní přijímat veškerý provoz na trunk lince. Následně odchytává VLAN tagované ethernetové rámce a kontroluje, jestli jsou definovány v tabulce přesměrování. Pokud ano, je VLAN ID přečíslováno podle této tabulky a celý rámec je prostřednictvím UDP paketu odeslán na danou ip adresu. Zde jej tunelovací server příjme, vybalí a opět v syrové formě ethernetového rámce a odešle ven trunk linkou. Tímto způsobem dochází k virtuálnímu propojení ethernetu definovaných VLAN a to i vzdáleně (přes Internet), tedy vznikne ethernetový tunel.

## 6.3 Virtlab - síťová komunikace

Virtlab z hlediskka síťové komunikace využívá při komunikaci s "vnějším světem"bezpečnostní protokol IPSec. IPSec protokol poskytuje vysoké zabezpečení, které je aplikováno z pohledu OSI modelu již na síťové vrstvě spojení.

IPsec tvoří logický tunel, který se navazuje mezi cílovou a zdrojovou ip, pro ostatní je tento tunel šifrován. V prostředí Virtlab je využíván jak protokol SA (Security Associations), tak protokol HA (Authentication Header),který zajišťuje autentizace odesilatele a příjemce, integritu dat v hlavičce, ale vlastní data nejsou šifrována, o šifrování dat se stará protokol ESP.

Mezi další používané prtokoly v prostředí Virtlab, patří klasické sí´tové prtokoly:

- HTTP Hypertext Transfer Protocol běžný internetový protokol komunikující v protokolu TCP na portu 80
- HTTPS Hypertext Transfer Protocol Secure je zabezpečená verze HTTP protokolu. Probíhající data jsou šifrována pomocí algroritmu TTL/TLS na straně serveru a komunikace probíhá na portu 443
- SSH Secure Shell je zabezpečený komunikační protokol, jedná se o náhradu ptozokolu telnet, který je z důvodu nízke bezpešnosti a snadného odposlouchávání komunikace, v prostředí virtlab nežádoucí. Jeho vysoká nebezpečnost je způsobena nešifrováním datové komunikace v tomto protokolu ( při odposlouchávání protokolu lze komunikace bezproblému zobrazit v plain textu). Z těchto důvodů je telnet nahrazen protokolem SSH komunikující na TCP portu 22.
- (NTP **[?]** NTP (Network Time Protocol) je protokol pro synchronizaci vnitřních hodin počítačů po paketové síti s proměnným zpožděním. Tento protokol zajišťuje, aby všechny počítače v síti měly stejný a přesný čas.
- SNMP Simple Network Management Protocol Slouží potřebám správy sítě. Umožouje průběžný sběr nejrůznějších dat pro potřeby správy sítě, a jejich nesledné vyhodnocování. Protokol je provozována UDP portu 161
- Další využívané protokoly pro vnitřní komunikaci v prostředí Virtlab Mezi další použité protokoly patří port 5666, a porty využité pro komunikaci s dalšími servervy v síti virtlab: port 10000, 10001, 40001, 50001, 50002, 60002.

# 7 Stavový firewall pro potřeby Virtlab

## 7.1 popis jednotlivých pravidel ve scriptu firewallu

```
configure_interfaces () {
    :
    # Configure interfaces
    update_addresses_of_interface "eth1 192.168.1.1/24" ""
    update_addresses_of_interface "lo 127.0.0.1/8"  ""
    getaddr eth0  i_eth0
    getaddr6 eth0  i_eth0_v6
}
```

Výpis 2: Nastavení základních parametrů

Nastavení ipadres a názvů jednotlivých síťových rozhraní na servru s firewallem, loopback je nastaven na defaultní adresu.

```
PATH="/sbin:/usr/sbin:/bin:/usr/bin:${PATH}"
export PATH

LSMOD="/sbin/lsmod"
MODPROBE="/sbin/modprobe"
IPTABLES="/sbin/iptables"
IP6TABLES="/sbin/ip6tables"
IPTABLES_RESTORE="/sbin/iptables-restore"
IP6TABLES_RESTORE="/sbin/ip6tables-restore"
IP="/sbin/ip"
IFCONFIG="/sbin/ifconfig"
VCONFIG="/sbin/vconfig"
BRCTL="/sbin/brctl"
IFENSLAVE="/sbin/ifenslave"
IPSET="/usr/sbin/ipset"
LOGGER="/usr/bin/logger"
```

Výpis 3: Nastavení základních parametrů

Načtení kernel modulů pro rozšířené funkce iptables. A nastavení cesty k programu iptables. Jednotlivé moduly pro zapnutí finkce logování v iptables, dále pak k zapnutí funkce REJECT - zahození paketu s odesláním icmp zprávy na původní zdrojovou adresu. Zavedení modulu pro zapnutí connection tracking, služící k vytvoření stavového paketového filtru.

Zapnutí rp filtru u důvodu ochrany proti ip spoofingu. IP spoofing znamená falšování zdrojové IP adresy, čímž se útočník snaží předstírat, že je někdo jiný, nebo se snaží zamaskovat svoji pravou IP adresu. Proto je žádoucí, aby se na vstupu firewallu filtrovaly pakety, které jsou evidentně podvržené a jejich původce nemůže mít čisté úmysly. Nejzákladnější ochranu před IP spoofingem představuje rpfilter. Jde o jednoduchou ochranu, která je vázána na konkrétní síťové rozhraní. Funguje tak, že jádro zablokuje pakety se zdrojovou adresou, která by podle routovací tabulky měla přijít z jiného dostupného rozhraní.

```
$IPTABLES -P INPUT ACCEPT
$IPTABLES -P OUTPUT ACCEPT
$IPTABLES -P FORWARD ACCEPT
```

Výpis 4: Nastavení politik firewallu

Nastavení politik u jednotlivých řetězců. Politika je pravidlo nastavující se u defaultních řetězců INPUT, OUTPUT, FORWARD, kterým se tyto řetězce řídí pokud žádné pravidlo v řetězci nevyhovuje žádané specifikaci.

```
# Rule 0 (eth0)
#
echo "Rule_0_(eth0)"
#
# anti spoofing log rule
$IPTABLES -N In_RULE_0
for i_eth0 in $i_eth0_list
do
test -n "$i_eth0" && $IPTABLES -A INPUT -i eth0 -s $i_eth0 -m limit --limit 5/hour --
    limit-burst 3 -j In_RULE_0
done
$IPTABLES -A INPUT -i eth0 -s 192.168.1.1 -m limit --limit 5/hour --limit-burst 3 -j
    In_RULE_0
$IPTABLES -A INPUT -i eth0 -s 10.0.0.0/8 -m limit --limit 5/hour --limit-burst 3 -j
    In_RULE_0
$IPTABLES -A INPUT -i eth0 -s 172.16.0.0/12 -m limit --limit 5/hour --limit-burst 3 -j
    In_RULE_0
$IPTABLES -A INPUT -i eth0 -s 192.168.0.0/16 -m limit --limit 5/hour --limit-burst 3 -j
    In_RULE_0
for i_eth0 in $i_eth0_list
do
test -n "$i_eth0" && $IPTABLES -A FORWARD -i eth0 -s $i_eth0 -m limit --limit 5/hour
    --limit-burst 3 -j In_RULE_0
done
$IPTABLES -A FORWARD -i eth0 -s 192.168.1.1 -m limit --limit 5/hour --limit-burst 3 -j
    In_RULE_0
$IPTABLES -A FORWARD -i eth0 -s 10.0.0.0/8 -m limit --limit 5/hour --limit-burst 3 -j
    In_RULE_0
$IPTABLES -A FORWARD -i eth0 -s 172.16.0.0/12 -m limit --limit 5/hour --limit-burst 3
    -j In_RULE_0
$IPTABLES -A FORWARD -i eth0 -s 192.168.0.0/16 -m limit --limit 5/hour --limit-burst 3
    -j In_RULE_0
$IPTABLES -A In_RULE_0 -j LOG --log-level warning --log-prefix "spoof"
```

Výpis 5: řetězec pro zprávu logování

Tento řetězec je ve firewalu obsažen z důvodu nutnosti logování průchodu jednotlivých paketů, avšak z důvodu velkého množství příchzích paketů se může log soubor zvětšit, až na nežádoucí velikost, proto díky rozšíření limit jsme omezily na zapsání prvních tří paketů do logu maximálně pětkrát do hodiny.

Dále je zde vidět vytvoření uživatelského řetězce, který se od defaultního liší tím, že se u nich nepoužívají polityky a spřehledoují a zlepšují škalovatelnost celého firewallu.

```
# Rule 1 (eth0)
#
echo "Rule 1 (eth0)"
#
# anti spoofing rule
for i_eth0 in $i_eth0_list
do
test -n "$i_eth0" && $IPTABLES -A INPUT -i eth0 -s $i_eth0 -j DROP
done
$IPTABLES -A INPUT -i eth0 -s 192.168.1.1 -j DROP
$IPTABLES -A INPUT -i eth0 -s 10.0.0.0/8 -j DROP
$IPTABLES -A INPUT -i eth0 -s 172.16.0.0/12 -j DROP
$IPTABLES -A INPUT -i eth0 -s 192.168.0.0/16 -j DROP
for i_eth0 in $i_eth0_list
do
test -n "$i_eth0" && $IPTABLES -A FORWARD -i eth0 -s $i_eth0 -j DROP
done
$IPTABLES -A FORWARD -i eth0 -s 192.168.1.1 -j DROP
$IPTABLES -A FORWARD -i eth0 -s 10.0.0.0/8 -j DROP
$IPTABLES -A FORWARD -i eth0 -s 172.16.0.0/12 -j DROP
$IPTABLES -A FORWARD -i eth0 -s 192.168.0.0/16 -j DROP
```

Výpis 6: řetězec pro ochranu proti spoofingu

V tomto řetězci pokračuje ochrana proti již výše zmiňovaného spoofingu, zde jsou zakázány adresy, které josu v internetu již registrovány pro jiné učely společností IANA a proto se v internetu rozshy těchto ip adres volně nepohybují a proto pokud se na síťovém rozhraní nekterá z těchto adres objeví lze tušit že se jedná o útok na naši síť. Všechny tyto adresy jsou logovány z důvodu pozdějšího dohledání.

```
# Povolení základních protokolů
$IPTABLES -A INPUT -i eth0 -p tcp -m tcp -m multiport --dports 22,53,80,443 -m state
    --state NEW -m connlimit \! --connlimit-above 50 -j ACCEPT
$IPTABLES -A INPUT -i eth0 -p udp -m udp -m multiport --dports 53,161,162 -m state
    --state NEW -m connlimit \! --connlimit-above 50 -j ACCEPT
$IPTABLES -A FORWARD -i eth0 -p tcp -m tcp -m multiport --dports 22,53,80,443 -m
    state --state NEW -m connlimit \! --connlimit-above 50 -j ACCEPT
$IPTABLES -A FORWARD -i eth0 -p udp -m udp -m multiport --dports 53,161,162 -m
    state --state NEW -m connlimit \! --connlimit-above 50 -j ACCEPT
```

Výpis 7: řetězec povolující porty pro vybrané protokoly

Zde jsou shrnuty pravidla pro povlení vybraných portů pro jednotlivé protokoly, v tomto případě se jedná o porty pro protokoly SSH, HTTP, HTTPS,které jsou používány a také SNMP sloužící pro sběr informací o daném síťovém prvku. pro komunikaci s prvky uvnitř sítě Virtlab.

```
# Povolení komunikace s jenotlivímy servery v prostředí virtlab
$IPTABLES -A INPUT -i eth0 -p tcp -m tcp -m multiport --dports
    10000,10001,60002,40001,50001,50002,60001,1000,60003,40002,5666 -m state --
    state NEW -m connlimit \! --connlimit-above 45 -j ACCEPT
```

```
$IPTABLES −A INPUT −i eth1 −p tcp −m tcp −m multiport −−dports
    10000,10001,60002,40001,50001,50002,60001,1000,60003,40002,5666 −m state −−
    state NEW −m connlimit \! −−connlimit−above 45 −j ACCEPT
$IPTABLES −A FORWARD −i eth0 −p tcp −m tcp −m multiport −−dports
    10000,10001,60002,40001,50001,50002,60001,1000,60003,40002,5666 −m state −−
    state NEW −m connlimit \! −−connlimit−above 45 −j ACCEPT
$IPTABLES −A FORWARD −i eth1 −p tcp −m tcp −m multiport −−dports
    10000,10001,60002,40001,50001,50002,60001,1000,60003,40002,5666 −m state −−
    state NEW −m connlimit \! −−connlimit−above 45 −j ACCEPT
$IPTABLES −A OUTPUT −o eth0 −p tcp −m tcp −m multiport −−dports
    10000,10001,60002,40001,50001,50002,60001,1000,60003,40002,5666 −m state −−
    state NEW −m connlimit \! −−connlimit−above 45 −j ACCEPT
$IPTABLES −A OUTPUT −o eth1 −p tcp −m tcp −m multiport −−dports
    10000,10001,60002,40001,50001,50002,60001,1000,60003,40002,5666 −m state −−
    state NEW −m connlimit \! −−connlimit−above 45 −j ACCEPT
$IPTABLES −A FORWARD −o eth0 −p tcp −m tcp −m multiport −−dports
    10000,10001,60002,40001,50001,50002,60001,1000,60003,40002,5666 −m state −−
    state NEW −m connlimit \! −−connlimit−above 45 −j ACCEPT
$IPTABLES −A FORWARD −o eth1 −p tcp −m tcp −m multiport −−dports
    10000,10001,60002,40001,50001,50002,60001,1000,60003,40002,5666 −m state −−
    state NEW −m connlimit \! −−connlimit−above 45 −j ACCEPT
```

Výpis 8: řetězec povolující porty pro potřeby sítě Virtlab

Tento řetězec obsahuje povolovací pravidla pro porty využívané při komunikaci s prvky virtlabu.

```
echo ”Rule 4 (eth0)”
#
# Povolení šifrovaného provozu
$IPTABLES −A INPUT −i eth0 −p 50 −m state −−state NEW −m connlimit \! −−connlimit−
    above 60 −j ACCEPT
$IPTABLES −A INPUT −i eth0 −p ah −m state −−state NEW −m connlimit \! −−connlimit−
    above 60 −j ACCEPT
$IPTABLES −A FORWARD −i eth0 −p 50 −m state −−state NEW −m connlimit \! −−
    connlimit−above 60 −j ACCEPT
$IPTABLES −A FORWARD −i eth0 −p ah −m state −−state NEW −m connlimit \! −−
    connlimit−above 60 −j ACCEPT
```

Výpis 9: řetězec povolující porty pro IPsec

Tyto pravidla povolují jednotlivé protokoly zabezpečení aby bylo možno vytvořit IPsec spojení kde protokol AH zajišťuje zabezpečení hlavičky paketu a protokol ESP se stará o šifrování dat v tomto paketu.

```
echo ”Rule 5 (eth0)”
#
# Povolení protokolů potřebných pro chod serverů
$IPTABLES −A INPUT −i eth0 −p icmp −m icmp −−icmp−type 0/0 −m state −−state NEW −
    m connlimit \! −−connlimit−above 10 −j ACCEPT
$IPTABLES −A INPUT −i eth0 −p icmp −m icmp −−icmp−type 8/0 −m state −−state NEW −
    m connlimit \! −−connlimit−above 10 −j ACCEPT
$IPTABLES −A INPUT −i eth0 −p udp −m udp −−dport 123 −m state −−state NEW −m
    connlimit \! −−connlimit−above 10 −j ACCEPT
```

```
$IPTABLES −A FORWARD −i eth0 −p icmp −m icmp −−icmp−type 0/0 −m state −−state
    NEW −m connlimit \! −−connlimit−above 10 −j ACCEPT
$IPTABLES −A FORWARD −i eth0 −p icmp −m icmp −−icmp−type 8/0 −m state −−state
    NEW −m connlimit \! −−connlimit−above 10 −j ACCEPT
$IPTABLES −A FORWARD −i eth0 −p udp −m udp −−dport 123 −m state −−state NEW −m
     connlimit \! −−connlimit−above 10 −j ACCEPT
```

Výpis 10: řetězec povolující porty pro méně užívané protokoly

V tomto řetězci jsou povoleny porty, které nesjou tak často volány jako například ntp protokol pro synchronizaci času.

```
echo "Rule 8 (eth0)"
#
# Ochrana před auth pakety
$IPTABLES −A INPUT −i eth0 −p tcp −m tcp −−dport 113 −j REJECT
$IPTABLES −A FORWARD −i eth0 −p tcp −m tcp −−dport 113 −j REJECT
```

Výpis 11: Ochrana proti AUTH protokolu

Některé programy používají protokol AUTH ke zjištění nežádoucích informací o firewallu, avšak při klasickém zahození paketu (DROP) se musí čekat na vypršení časového limitu a tím se spomaluje komunikace. K odstranění tohoto problému je třeba využít akci REJECT, k odeslání icmp zprávy zpět na cílovou adresu. Samozřejmě všechny pokusi o zjištění slabin sítě jsou logovány.

```
$IPTABLES −A INPUT −i eth0 −p icmp −m icmp −−icmp−type 0/0 −m state −−state NEW −m
    connlimit \! −−connlimit−above 10 −j ACCEPT
$IPTABLES −A INPUT −i eth0 −p icmp −m icmp −−icmp−type 8/0 −m state −−state NEW −
    m connlimit \! −−connlimit−above 10 −j ACCEPT
$IPTABLES −A FORWARD −i eth0 −p icmp −m icmp −−icmp−type 0/0 −m state −−state
    NEW −m connlimit \! −−connlimit−above 10 −j ACCEPT
$IPTABLES −A FORWARD −i eth0 −p icmp −m icmp −−icmp−type 8/0 −m state −−state
    NEW −m connlimit \! −−connlimit−above 10 −j ACCEPT
```

Výpis 12: Omezení typů ICMP zpráv

Zde je omezení ICMP zpráv pouze na zprávu ping, pomocí které se měří doba odezvy od vzdáleného síťového rozhraní.

```
echo "Rule 2 (lo)"
#
# Povolení přístupu na loopback
$IPTABLES −A INPUT −i lo −m state −−state NEW −j ACCEPT
$IPTABLES −A OUTPUT −o lo −m state −−state NEW −j ACCEPT
```

Výpis 13: Povolení provozu na virtualním rozhraní systému

Povolení provozu na vnitřním virtuálním síťovém rozhraní systému.

```
echo "Rule 9 (eth1)"
#
# Povolení broadcastů z vnitřní sítě
$IPTABLES −A OUTPUT −o eth1 −d 255.255.255.255 −m state −−state NEW −j ACCEPT
```

Výpis 14: Povolení broadcastu z vnitřní sítě

V tomto případě jsou povoleny broadcasty z vnitřní sítě virtlabu, které je zabezpečená a proto můžeme broadcast pakety povolit.

```
$IPTABLES -A INPUT -d -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Výpis 15: Povolení broadcastu z vnitřní sítě

U již vytvořených a provozovaných spojení vímže, že již prošli přez paketový filter a je proto bezpečné tato spojení pomocí stavového paketového filtru povolit.

```
    # Logovací pravidlo zahozených paketů
    $IPTABLES -N RULE_10
    $IPTABLES -A OUTPUT -m limit --limit 12/hour --limit-burst 5 -j RULE_10
    $IPTABLES -A INPUT -m limit --limit 12/hour --limit-burst 5 -j RULE_10
    $IPTABLES -A FORWARD -m limit --limit 12/hour --limit-burst 5 -j RULE_10
    $IPTABLES -A RULE_10 -j LOG --log-level info --log-prefix "RULE 10 -- CONTINUE
        "
    #
    # Rule 11 (global)
    #
    echo "Rule 11 (global)"
    #
    # Obecné pravidlo pro zahození nevyhovujících paketů
    $IPTABLES -A OUTPUT -j DROP
    $IPTABLES -A INPUT -j DROP
    $IPTABLES -A FORWARD -j DROP
}
```

Výpis 16: Zahození veškerého nevyhovujícího provozu

Zde je poslední pravidlo ve firewallu, které vzhledem k nastavení politiky v řetězci INPUT na ACCEPT je nutné veškerý ostatní síťový provoz nepropusti skrze firewall. Samozřejmě se tyto zahozené pakety budou logovat, spříslošným příznakem. Vzhledek k zajištění určitého počtu logů je pravidlo nastaveno pro vytvoření maximálně dvanácti logů za hodinu.

## 7.2 Kompletní script pro vytvoření firewallu

```
#!/bin/sh
#
#  This is automatically generated file. DO NOT MODIFY !
#
#  Firewall Builder  fwb_ipt v4.2.0.3530
#
#  Generated Fri Jul 22 12:45:50 2011 St?edn? Evropa (letn? ?as) by Admin
#
#  files : * Virtlab.fw /etc/Virtlab.fw
#
# Compiled for iptables (any version)
#
# This firewall has two interfaces. Eth0 faces outside and has a dynamic address; eth1 faces
    inside.
```

```
# Policy includes basic rules to permit unrestricted outbound access and anti-spoofing rules.
      Access to the firewall is permitted only from internal network and only using SSH. The
      firewall uses one of the machines on internal network for DNS. Internal network is configured
      with address 192.168.1.0/255.255.255.0



set -x
FWBDEBUG=""

PATH="/sbin:/usr/sbin:/bin:/usr/bin:${PATH}"
export PATH



LSMOD="lsmod"
MODPROBE="modprobe"
IPTABLES="iptables"
IP6TABLES="ip6tables"
IPTABLES_RESTORE="iptables-restore"
IP6TABLES_RESTORE="ip6tables-restore"
IP="ip"
IFCONFIG="ifconfig"
VCONFIG="vconfig"
BRCTL="brctl"
IFENSLAVE="ifenslave"
IPSET="ipset"
LOGGER="logger"

log() {
    echo "$1"
    command -v "$LOGGER" >/dev/null 2>&1 && $LOGGER -p info "$1"
}

getInterfaceVarName() {
    echo $1 | sed 's/\./_/'
}

getaddr_internal() {
    dev=$1
    name=$2
    af=$3
    L=$($IP $af addr show dev $dev | sed -n '/inet/{s!.*inet6* !!; s !/.*!! p}' | sed 's/peer.*// ')
    test -z "$L" && {
        eval "$name=''"
        return
    }
    eval "${name}_list=\"$L\""
}

getaddr() {
    getaddr_internal $1 $2 "-4"
}
```

```
getaddr6() {
    getaddr_internal $1 $2 "-6"
}

# function getinterfaces is used to process wildcard interfaces
getinterfaces () {
    NAME=$1
    $IP link show | grep ": $NAME" | while read L; do
        OIFS=$IFS
        IFS=" :"
        set $L
        IFS=$OIFS
        echo $2
    done
}

diff_intf () {
    func=$1
    list1 =$2
    list2 =$3
    cmd=$4
    for intf in $list1
    do
        echo $list2 | grep -q $intf || {
        # $vlan is absent in list 2
            $func $intf $cmd
        }
    done
}

find_program() {
  PGM=$1
  command -v $PGM >/dev/null 2>&1 || {
    echo "$PGM not found"
    exit 1
  }
}
check_tools() {
  find_program $IPTABLES
 find_program $MODPROBE
  find_program $IP
}
reset_iptables_v4 () {
  $IPTABLES -P OUTPUT DROP
  $IPTABLES -P INPUT DROP
  $IPTABLES -P FORWARD DROP

cat /proc/net/ip_tables_names | while read table; do
  $IPTABLES -t $table -L -n | while read c chain rest; do
      if test "X$c" = "XChain" ; then
        $IPTABLES -t $table -F $chain
      fi
  done
  $IPTABLES -t $table -X
```

```
done
}

reset_iptables_v6 ()  {
  $IP6TABLES -P OUTPUT DROP
  $IP6TABLES -P INPUT DROP
  $IP6TABLES -P FORWARD DROP

cat  /proc/net/ip6_tables_names | while read table;  do
  $IP6TABLES -t $table -L -n | while read c chain rest; do
       if  test  "X$c" = "XChain" ; then
         $IP6TABLES -t $table -F $chain
       fi
  done
  $IP6TABLES -t $table -X
done
}


P2P_INTERFACE_WARNING=""

missing_address() {
     address=$1
     cmd=$2

     oldIFS=$IFS
     IFS="@"
     set  $address
     addr=$1
     interface=$2
     IFS=$oldIFS



     $IP addr show dev $interface | grep -q POINTOPOINT && {
          test  -z "$P2P_INTERFACE_WARNING" && echo "Warning: Can not update address of
               interface $interface. fwbuilder can not manage addresses of point-to-point
               interfaces yet"
          P2P_INTERFACE_WARNING="yes"
          return
     }

     test  "$cmd" = "add" && {
       echo "# Adding ip address: $interface $addr"
       echo $addr | grep -q  ':'  && {
            $FWBDEBUG $IP addr $cmd $addr dev $interface
       }  ||  {
            $FWBDEBUG $IP addr $cmd $addr broadcast + dev $interface
       }
     }

     test  "$cmd" = "del" && {
       echo "# Removing ip address: $interface $addr"
       $FWBDEBUG $IP addr $cmd $addr dev $interface || exit 1
```

```
    }

    $FWBDEBUG $IP link set $interface up
}

list_addresses_by_scope() {
    interface=$1
    scope=$2
     ignore_list =$3
    $IP addr ls  dev $interface  | \
      awk -v IGNORED="$ignore_list" -v SCOPE="$scope" \
        'BEGIN {
            split (IGNORED,ignored_arr);
           for  (a in  ignored_arr)  {ignored_dict[ignored_arr[a]]=1;}
         }
         (/ inet  |inet6  /  && $0 ~ SCOPE && !($2 in ignored_dict)) {print $2;}'  |  \
        while read addr; do
          echo "${addr}@$interface"
   done | sort
}


update_addresses_of_interface() {
     ignore_list =$2
    set  $1
    interface=$1
     shift

    FWB_ADDRS=$(
      for  addr in  $*; do
        echo "${addr}@$interface"
      done | sort
    )

    CURRENT_ADDRS_ALL_SCOPES=""
    CURRENT_ADDRS_GLOBAL_SCOPE=""

    $IP  link  show dev $interface >/dev/null 2>&1 && {
      CURRENT_ADDRS_ALL_SCOPES=$(list_addresses_by_scope $interface 'scope .*' "
          $ignore_list")
      CURRENT_ADDRS_GLOBAL_SCOPE=$(list_addresses_by_scope $interface 'scope global' "
          $ignore_list")
    } ||  {
      echo "# Interface $interface does not exist"
      # Stop the  script  if  we are not in  test  mode
      test  -z "$FWBDEBUG" && exit 1
    }

     diff_intf   missing_address "$FWB_ADDRS" "$CURRENT_ADDRS_ALL_SCOPES" add
     diff_intf   missing_address "$CURRENT_ADDRS_GLOBAL_SCOPE" "$FWB_ADDRS" del
}

clear_addresses_except_known_interfaces() {
    $IP  link  show | sed 's/: //g'  |  awk -v IGNORED="$*" \
```

```
        'BEGIN {
            split (IGNORED,ignored_arr);
           for (a in ignored_arr) {ignored_dict[ignored_arr[a]]=1;}
         }
         (/state/ && !($2 in ignored_dict)) {print $2;}' | \
         while read intf ; do
            echo "# Removing addresses not configured in fwbuilder from interface $intf"
            $FWBDEBUG $IP addr flush dev $intf scope global
            $FWBDEBUG $IP link set $intf down
         done
}

check_file () {
    test -r "$2" || {
        echo "Can not find file $2 referenced by address table object $1"
        exit 1
    }
}

check_run_time_address_table_files () {
    :

}

load_modules() {
    :
    OPTS=$1
    MODULES_DIR="/lib/modules/`uname -r`/kernel/net/"
    MODULES=$(find $MODULES_DIR -name '*conntrack*' \! -name '*ipv6*'|sed -e 's/^.*\///' -
        e 's/\([^\.]\)\..*/\1/')
    echo $OPTS | grep -q nat && {
        MODULES="$MODULES $(find $MODULES_DIR -name '*nat*'|sed -e 's/^.*\///' -e 's
            /\([^\.]\)\..*/\1/')"
    }
    echo $OPTS | grep -q ipv6 && {
        MODULES="$MODULES $(find $MODULES_DIR -name nf_conntrack_ipv6|sed -e 's
            /^.*\///' -e 's/\([^\.]\)\..*/\1/')"
    }
    for module in $MODULES; do
        if $LSMOD | grep ${module} >/dev/null; then continue; fi
        $MODPROBE ${module} || exit 1
    done
}

verify_interfaces () {
    :
    echo "Verifying interfaces: eth0 eth1 lo"
    for i in eth0 eth1 lo ; do
        $IP link show "$i" > /dev/null 2>&1 || {
            log "Interface $i does not exist"
            exit 1
        }
    done
}
```

```
prolog_commands() {
    echo "Running prolog script"

}

epilog_commands() {
    echo "Running epilog script"

}

run_epilog_and_exit() {
    epilog_commands
    exit $1
}

configure_interfaces() {
    :
    # Configure interfaces
    update_addresses_of_interface "eth1 192.168.1.1/24" ""
    update_addresses_of_interface "lo 127.0.0.1/8" ""
    getaddr eth0 i_eth0
    getaddr6 eth0 i_eth0_v6
}

script_body() {
    # ================ IPv4


    # ================ Table 'filter', automatic rules
    # accept established sessions
    $IPTABLES -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
    $IPTABLES -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
    $IPTABLES -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT


    # ================ Table 'nat', rule set NAT
    #
    # Rule 0 (NAT)
    #
    echo "Rule 0 (NAT)"
    #
    $IPTABLES -t nat -A POSTROUTING -o eth0 -s 192.168.1.0/24 -j MASQUERADE



    # ================ Table 'filter', rule set Policy
    #
    # Rule 0 (lo)
    #
    echo "Rule 0 (lo)"
    #
    # Povolení přístupu na loopback
    $IPTABLES -A INPUT -i lo -m state --state NEW -j ACCEPT
```

```
$IPTABLES −A OUTPUT −o lo −m state −−state NEW −j ACCEPT
#
# Rule 1 (eth0)
#
echo "Rule 1 (eth0)"
#
# Povolení šifrovaného provozu
$IPTABLES −A INPUT −i eth0 −p 50 −m state −−state NEW −m connlimit \! −−connlimit−
    above 60 −j ACCEPT
$IPTABLES −A INPUT −i eth0 −p ah −m state −−state NEW −m connlimit \! −−connlimit−
    above 60 −j ACCEPT
$IPTABLES −A FORWARD −i eth0 −p 50 −m state −−state NEW −m connlimit \! −−
    connlimit−above 60 −j ACCEPT
$IPTABLES −A FORWARD −i eth0 −p ah −m state −−state NEW −m connlimit \! −−
    connlimit−above 60 −j ACCEPT
#
# Rule 2 (eth0)
#
echo "Rule 2 (eth0)"
#
# Povolení základních protokolů
$IPTABLES −A INPUT −i eth0 −p tcp −m tcp −m multiport −−dports 53,80,443,22 −m state
    −−state NEW −m connlimit \! −−connlimit−above 50 −j ACCEPT
$IPTABLES −A INPUT −i eth0 −p udp −m udp −m multiport −−dports 53,161,162 −m state
    −−state NEW −m connlimit \! −−connlimit−above 50 −j ACCEPT
$IPTABLES −A FORWARD −i eth0 −p tcp −m tcp −m multiport −−dports 53,80,443,22 −m
    state −−state NEW −m connlimit \! −−connlimit−above 50 −j ACCEPT
$IPTABLES −A FORWARD −i eth0 −p udp −m udp −m multiport −−dports 53,161,162 −m
    state −−state NEW −m connlimit \! −−connlimit−above 50 −j ACCEPT
#
# Rule 3 (eth0,eth1)
#
echo "Rule 3 (eth0,eth1)"
#
# Povolení komunikace s jenotlivímy servery v prostředí  virtlab
$IPTABLES −A INPUT −i eth0 −p tcp −m tcp −m multiport −−dports
    10001,60001,1000,60002,60003,50001,40001,40002,50002,5666,10000 −m state −−
    state NEW −m connlimit \! −−connlimit−above 45 −j ACCEPT
$IPTABLES −A INPUT −i eth1 −p tcp −m tcp −m multiport −−dports
    10001,60001,1000,60002,60003,50001,40001,40002,50002,5666,10000 −m state −−
    state NEW −m connlimit \! −−connlimit−above 45 −j ACCEPT
$IPTABLES −A FORWARD −i eth0 −p tcp −m tcp −m multiport −−dports
    10001,60001,1000,60002,60003,50001,40001,40002,50002,5666,10000 −m state −−
    state NEW −m connlimit \! −−connlimit−above 45 −j ACCEPT
$IPTABLES −A FORWARD −i eth1 −p tcp −m tcp −m multiport −−dports
    10001,60001,1000,60002,60003,50001,40001,40002,50002,5666,10000 −m state −−
    state NEW −m connlimit \! −−connlimit−above 45 −j ACCEPT
$IPTABLES −A OUTPUT −o eth0 −p tcp −m tcp −m multiport −−dports
    10001,60001,1000,60002,60003,50001,40001,40002,50002,5666,10000 −m state −−
    state NEW −m connlimit \! −−connlimit−above 45 −j ACCEPT
$IPTABLES −A OUTPUT −o eth1 −p tcp −m tcp −m multiport −−dports
    10001,60001,1000,60002,60003,50001,40001,40002,50002,5666,10000 −m state −−
    state NEW −m connlimit \! −−connlimit−above 45 −j ACCEPT
```

```
$IPTABLES −A FORWARD −o eth0 −p tcp −m tcp −m multiport −−dports
    10001,60001,1000,60002,60003,50001,40001,40002,50002,5666,10000 −m state −−
    state NEW −m connlimit \! −−connlimit−above 45 −j ACCEPT
$IPTABLES −A FORWARD −o eth1 −p tcp −m tcp −m multiport −−dports
    10001,60001,1000,60002,60003,50001,40001,40002,50002,5666,10000 −m state −−
    state NEW −m connlimit \! −−connlimit−above 45 −j ACCEPT
#
# Rule 4 (eth0)
#
echo ”Rule␣4␣(eth0)”
#
# Povolení protokolů potřebných pro chod serverů
$IPTABLES −A INPUT −i eth0 −p icmp −m icmp −−icmp−type 0/0 −m state −−state NEW −
    m connlimit \! −−connlimit−above 10 −j ACCEPT
$IPTABLES −A INPUT −i eth0 −p icmp −m icmp −−icmp−type 8/0 −m state −−state NEW −
    m connlimit \! −−connlimit−above 10 −j ACCEPT
$IPTABLES −A INPUT −i eth0 −p udp −m udp −−dport 123 −m state −−state NEW −m
    connlimit \! −−connlimit−above 10 −j ACCEPT
$IPTABLES −A FORWARD −i eth0 −p icmp −m icmp −−icmp−type 0/0 −m state −−state
    NEW −m connlimit \! −−connlimit−above 10 −j ACCEPT
$IPTABLES −A FORWARD −i eth0 −p icmp −m icmp −−icmp−type 8/0 −m state −−state
    NEW −m connlimit \! −−connlimit−above 10 −j ACCEPT
$IPTABLES −A FORWARD −i eth0 −p udp −m udp −−dport 123 −m state −−state NEW −m
     connlimit \! −−connlimit−above 10 −j ACCEPT
#
# Rule 5 (eth1)
#
echo ”Rule␣5␣(eth1)”
#
# Povolení broadcastů z vnitřní sítě
$IPTABLES −A OUTPUT −o eth1 −d 255.255.255.255 −m state −−state NEW −j ACCEPT
#
# Rule 6 (eth0)
#
echo ”Rule␣6␣(eth0)”
#
# Logovací pravidlo pro ochranu před AUTH pakety
$IPTABLES −A INPUT −i eth0 −p tcp −m tcp −−dport 113 −j LOG −−log−level info −−log−
    prefix ”RULE␣6␣−−␣CONTINUE␣”
$IPTABLES −A FORWARD −i eth0 −p tcp −m tcp −−dport 113 −j LOG −−log−level info −−
    log−prefix ”RULE␣6␣−−␣CONTINUE␣”
#
# Rule 7 (eth0)
#
echo ”Rule␣7␣(eth0)”
#
# Ochrana před auth pakety
$IPTABLES −A INPUT −i eth0 −p tcp −m tcp −−dport 113 −j REJECT
$IPTABLES −A FORWARD −i eth0 −p tcp −m tcp −−dport 113 −j REJECT
#
# Rule 8 (eth0)
#
echo ”Rule␣8␣(eth0)”
#
```

```
# anti spoofing log rule
for i_eth0 in $i_eth0_list
do
test -n "$i_eth0" && $IPTABLES -A INPUT -i eth0 -s $i_eth0 -j LOG --log-level warning
    --log-prefix "spoof"
done
$IPTABLES -A INPUT -i eth0 -s 192.168.1.1 -j LOG --log-level warning --log-prefix "
    spoof"
$IPTABLES -A INPUT -i eth0 -s 10.0.0.0/8 -j LOG --log-level warning --log-prefix "
    spoof"
$IPTABLES -A INPUT -i eth0 -s 172.16.0.0/12 -j LOG --log-level warning --log-prefix "
    spoof"
$IPTABLES -A INPUT -i eth0 -s 192.168.0.0/16 -j LOG --log-level warning --log-prefix
    "spoof"
for i_eth0 in $i_eth0_list
do
test -n "$i_eth0" && $IPTABLES -A FORWARD -i eth0 -s $i_eth0 -j LOG --log-level
    warning --log-prefix "spoof"
done
$IPTABLES -A FORWARD -i eth0 -s 192.168.1.1 -j LOG --log-level warning --log-
    prefix "spoof"
$IPTABLES -A FORWARD -i eth0 -s 10.0.0.0/8 -j LOG --log-level warning --log-prefix
    "spoof"
$IPTABLES -A FORWARD -i eth0 -s 172.16.0.0/12 -j LOG --log-level warning --log-
    prefix "spoof"
$IPTABLES -A FORWARD -i eth0 -s 192.168.0.0/16 -j LOG --log-level warning --log-
    prefix "spoof"
#
# Rule 9 (eth0)
#
echo "Rule 9 (eth0)"
#
# anti spoofing rule
for i_eth0 in $i_eth0_list
do
test -n "$i_eth0" && $IPTABLES -A INPUT -i eth0 -s $i_eth0 -j DROP
done
$IPTABLES -A INPUT -i eth0 -s 192.168.1.1 -j DROP
$IPTABLES -A INPUT -i eth0 -s 10.0.0.0/8 -j DROP
$IPTABLES -A INPUT -i eth0 -s 172.16.0.0/12 -j DROP
$IPTABLES -A INPUT -i eth0 -s 192.168.0.0/16 -j DROP
for i_eth0 in $i_eth0_list
do
test -n "$i_eth0" && $IPTABLES -A FORWARD -i eth0 -s $i_eth0 -j DROP
done
$IPTABLES -A FORWARD -i eth0 -s 192.168.1.1 -j DROP
$IPTABLES -A FORWARD -i eth0 -s 10.0.0.0/8 -j DROP
$IPTABLES -A FORWARD -i eth0 -s 172.16.0.0/12 -j DROP
$IPTABLES -A FORWARD -i eth0 -s 192.168.0.0/16 -j DROP
#
# Rule 10 (global)
#
echo "Rule 10 (global)"
#
```

```
    # Logovací pravidlo zahozených paketů
    $IPTABLES -A OUTPUT -j LOG --log-level info --log-prefix "RULE 10 -- CONTINUE
        "
    $IPTABLES -A INPUT -j LOG --log-level info --log-prefix "RULE 10 -- CONTINUE "
    $IPTABLES -A FORWARD -j LOG --log-level info --log-prefix "RULE 10 --
        CONTINUE "
    #
    # Rule 11 (global)
    #
    echo "Rule 11 (global)"
    #
    # Obecné pravidlo pro zahození nevyhovujících paketů
    $IPTABLES -A OUTPUT -j DROP
    $IPTABLES -A INPUT -j DROP
    $IPTABLES -A FORWARD -j DROP
}

ip_forward() {
    :
    echo 1 > /proc/sys/net/ipv4/ip_forward
}

reset_all () {
    :
    reset_iptables_v4
}

block_action() {
    reset_all
}

stop_action() {
    reset_all
    $IPTABLES -P OUTPUT ACCEPT
    $IPTABLES -P INPUT ACCEPT
    $IPTABLES -P FORWARD ACCEPT
}

check_iptables() {
    IP_TABLES="$1"
    [ ! -e $IP_TABLES ] && return 151
    NF_TABLES=$(cat $IP_TABLES 2>/dev/null)
    [ -z "$NF_TABLES" ] && return 152
    return 0
}
status_action() {
    check_iptables "/proc/net/ip_tables_names"
    ret_ipv4=$?
    check_iptables "/proc/net/ip6_tables_names"
    ret_ipv6=$?
    [ $ret_ipv4 -eq 0 -o $ret_ipv6 -eq 0 ] && return 0
    [ $ret_ipv4 -eq 151 -o $ret_ipv6 -eq 151 ] && {
        echo "iptables modules are not loaded"
    }
```

```
    [ $ret_ipv4 -eq 152 -o $ret_ipv6 -eq 152 ] && {
        echo "Firewall is not configured"
    }
    exit 3
}

# See how we were called.
# For backwards compatibility missing argument is equivalent to 'start'

cmd=$1
test -z "$cmd" && {
    cmd="start"
}

case "$cmd" in
    start)
        log "Activating  firewall  script generated Fri Jul 22 12:45:50 2011 by Admin"
        check_tools
         prolog_commands
        check_run_time_address_table_files

        load_modules "nat "
        configure_interfaces
         verify_interfaces

          reset_all

        script_body
        ip_forward
        epilog_commands
        RETVAL=$?
        ;;

    stop)
        stop_action
        RETVAL=$?
        ;;

    status)
        status_action
        RETVAL=$?
        ;;

    block)
        block_action
        RETVAL=$?
        ;;

    reload)
        $0 stop
        $0 start
        RETVAL=$?
        ;;
```

```
    interfaces)
        configure_interfaces
        RETVAL=$?
        ;;

    test_interfaces)
        FWBDEBUG="echo"
        configure_interfaces
        RETVAL=$?
        ;;



    *)
        echo "Usage: $0 [start|stop|status|block|reload|interfaces|test_interfaces]"
        ;;

esac

exit $RETVAL
```

Výpis 17: Kompletní script pro vytvoření firewallu

# 8 Závěr

Cílem této práce bylo vytvoření stavového paketového filtru, který bude nasazen a je přizbůsoben pro prostředí Virtlab. Při dosahování toho cíle jsem se hodně dozvěděl nejen o fungování firewallu a iptables, ale také o samotném fungování jednotlivých protokolů.

Tyto poznatky jsem využil při konfiguraci paketového filtru, v síťovém prostředí na platformě Mikrotik. Kde jsem konstruoval firewall pro klieantské zařízení, poskytovatele bezdrátového internetu.

U tohoto firewallu vidím slabé místo v nedostatečném filtrování protokolu IPsec, jelikož do tohoto šifrovaného protokolu nemůže firewall nahlížet, musí "věřit", že na druhé straně tunelu je důvěryhodný zdroj. Problém by se dal vyřešit podobným způsobe napodobující útok typu "stand in the middle"kde by se firewal choval jako cíl vytvořeného tunelu a zde přebíral data, které zdroj posílá, ty pak filtrovat a posílat původnímu cíli již přefiltrovaný provoz. Cílový prvek by si myslel, že komunikuje přímo se zdrojem, ale místo toho by komunikoval pouze s firewallem. tuto techniku využívá například společnost microsoft u svých Threat Management Gateway.

# 9 Reference

[1] *Úvodem - Virtuální laboratoř počítačových sítí* [online]. 7. 10. 2010 [cit. 2011-04-10]. Virtuální laboratoř počítačových sítí. Dostupné z WWW: http://infra2.cs.vsb.cz/vl-wiki/index.php/Virtuální$_l aborato\check{r}_p očítačových_s ítí$

[2] *Historie - Virtuální laboratoř počítačových sítí* [online]. 7. 10. 2010 [cit. 2011-04-010]. Virtuální laboratoř počítačových sítí. Dostupné z WWW: http://infra2.cs.vsb.cz/vl-wiki/index.php/Virtuální$_l aborato\check{r}_p očítačových_s ítí$

[3] *Firewall. In Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 18. 11. 2007, last modified on 22. 4. 2011 [cit. 2011-04-10]. Dostupné z WWW: http://cs.wikipedia.org/wiki/Firewall.

[4] *NTP. In Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 17. 6. 2005, last modified on 15. 11. 2010 [cit. 2011-04-10]. Dostupné z WWW: http://cs.wikipedia.org/wiki/NTP.

[5] *Rezervační server - Virtuální laboratoř počítačových sítí* [online]. 7. 10. 2010 [cit. 2011-04-010]. Virtuální laboratoř počítačových sítí. Dostupné z WWW: http://infra2.cs.vsb.cz/vl-wiki/index.php/Virtlab:Komponenty/Rezerva

[6] *Mazací server - Virtuální laboratoř počítačových sítí* [online]. 7. 10. 2010 [cit. 2011-04-010]. Virtuální laboratoř počítačových sítí. Dostupné z WWW: http://infra2.cs.vsb.cz/vl-wiki/index.php/Virtlab:Komponenty/Mazac

[7] *Konzolový server - Virtuální laboratoř počítačových sítí* [online]. 7. 10. 2010 [cit. 2011-04-010]. Virtuální laboratoř počítačových sítí. Dostupné z WWW: http://infra2.cs.vsb.cz/vl-wiki/index.php/Virtlab:Konzolov

[8] *Tunelovací server - Virtuální laboratoř počítačových sítí* [online]. 7. 10. 2010 [cit. 2011-04-010]. Virtuální laboratoř počítačových sítí. Dostupné z WWW: http://infra2.cs.vsb.cz/vl-wiki/index.php/Virtlab:Tunelovac